Lowell Batacan
Joshua Steward
Jessica Perez
Zachery Van Es
Zane Gittins
Andrew Walters

Epidemiology Group #8
Non-functional Requirements

**(1). Usability:**

Goal: The program should be easy to use for people who are familiar with computers and games. It should be easy to start and stop the program as well as be able to navigate through the menus within the program. It should also provide a challenge and draw out critical thinking and strategic thinking from the user, while still remaining usable and maintaining a low learning curve.

(a) validity: To improve how easy it is to use does support the customers needs.

(b) consistency: The ease of use does not conflict with other requirements. In fact, being accessible and usable should bolster our environmental and operational requirements as discussed later on.

(c) completeness: Yes all requirements are there. We have met all the goals set out in our plans, have documented our work extensively, and even plan to expand on our work.

(d) realism: implementing this kind of user friendly system should be possible within the allotted time. We have also discussed how to split up the work based on expertise in specific areas to maximize productivity. Scrum will also allow a consistent and efficient use of time that will ensure a maximization of productivity and team-centric focus.

(e) verifiability: The ease of use can be tested; anyone can play on multiple systems. Builds will be put out for multiple operating systems and platforms. Scrum frameworks will allow teams to verify progress on usability. An easy method will be to check for total compilation during each daily scrum.

(f) comprehensibility: It is very easy to understand the requirements put forth, and there is no previous knowledge required to use the software. There will also be a short learning guide to guide you through basic controls, along with a help page.

(g) traceability: In the original system specification it was decided that our program would have a simple and easy to use interface, along with logged debugging code to debug in-editor.

(h) adaptability: It should be fairly adaptable to multiple platforms without affecting other requirements. Platforms and APIs used were picked for just this purpose; adaptability and transportability.

**(2). Performance:**

Goal: The program should run well on all platforms it is used on, it should not have framerate issues, and it should not require a dedicated graphical processor. The program should run well even when there are large crowds of people on screen.

(a) validity: It does perform functions which best support the customers needs in terms of performance.

(b) consistency: No there are no requirement conflicts.

(c) completeness: Yes all functions required by the customer in terms of performance are included.

(d) realism: Yes the performance requirement can be implemented given the available budget and technology. We are building are project in Unity3D which allows for ports to multiple systems easily, as well as the pathfinding system we are using is optimized for high performance with multiple actors on screen.

(e) verifiability: Yes this requirement can be checked by seeing how well it performs on multiple computers, specifically by measuring how many times the program stutters or lags. If it does not stutter or lag more than a few times each use then it succeeds

(f) comprehensibility: Yes this requirement is properly understood by the team. We understand that we need to optimize code to allow for this best performance.

(g) traceability: The origin of the requirement is that the team wished for the program to run effectively on multiple systems.

(h) adaptability: No the requirement cannot be changed without affecting multiple systems because if the performance requirement is changed then it will affect the dependability of the program.

**(3). Space:**

Goal: The program should fit on a personal computer and take up reasonable space for a executable program. It should be fairly compact and portable so that it can be widely used by many different systems. In terms of final space, the final build of the game will also be much smaller than the project itself. This is because the framework uses many external files to build projects and assist in project development, but when the final build of the project is built, unneeded elements are essentially cut out of the project, and an executable is built out of it.

(a) validity: The small space would support customers needs on most platforms, and make the experience more enjoyable as well as support other non-functional requirements, including usability.

(b) consistency: This shouldn't conflict with other requirements unless another one needs more space than allotted.

(c) completeness: All functions should be included, along with documentation. These will use outside APIs that have been developed and tested for their low-overhead memory requirements.

(d) realism: This will be easy to do as it should be a side-effect of the other requirements.

(e) verifiability: It is also the easiest requirement to check as it will be number that can be checked.

(f) comprehensibility: The requirement is easy to understand as the concept is simple.

(g) traceability: The team has an idea of the end size of the final program.

(h) adaptability: The size can be changed depending on new features that may be added later.

**(4). Dependability:**

Goal: The program should run and not crash no matter what platform you play it on. As long as the system requirements are met then the game should run smoothly. Un-rigorous system requirements will also be set forth; we are using fairly basic graphical elements, no particle effects, and lots of base level, efficient code.

(a) validity: Yes, the program will be tested multiple times to make sure it will not crash and run smoothly.

(b) consistency: No, there is no conflict between dependability and others on this list.

(c) completeness: Yes it will have functions that will keep it from crashing while running. Many of these will be built in type-safe catchers to catch logs of object instances that run in loops that don't converge.

(d) realism: With the time and budget it is possible to to complete this requirement though it might be hard within the time frame to find every bug.

(e) verifiability: Yes this requirement can be checked through multiple tests of the program and trying actions in the program that a user might try in the game.

(f) comprehensibility: Yes this requirement is understood. If the dependability of the program is low then it will fail often leaving the user frustrated.

(g) traceability: The origin of this requirement is the program itself. If the program fails running normally then the dependability is called into question. Also looking for bugs in the program even though the program looks like it is running normally is another point towards it origin that is the program itself.

(h) adaptability: The more dependable it is the less features that might be offered in the program. The more features you add to a program the less stable it can become since it allows the user to alter more variable/objects/ or events in the program.

**(5). Environmental:**

Goal: The program should run on most PC operating systems such as Windows, Mac, and linux. It will be a executable program that run in a closed environment. It should be easy to move the program from different systems.

(a) validity:It should support any customer's needs as it can work on most operating systems

(b) consistency: There are no known conflicts

(c) completeness: Yes, all functions should be available.

(d) realism: We should be able to have all available versions because of our development process.

(e) verifiability: Yes by running this program in different environments we can test and make sure the program will run.

(f) comprehensibility: The need for use on many systems is fairly easy to understand.

(g) traceability: Yes this origin can be traced.

(h) adaptability: Changing this would most likely affect other requirements as the system that our program operates on effects how to code it.

**(6). Operational:**

Goal: The product should be complete and usable by its launch date. The product should also be ready for further improvements and evolution. The development team should be ready for any deficiency and enhance the overall product.

(a) validity: It is possible to debug any errors and prepare a process to handle upcoming improvements and implementations. In terms of operational language debugging, we will be using object oriented visual studio debugging which will allow us to go through large data-structures and debug quickly and effectively.

(b) consistency: As long as the evolution process doesn't change the overall functionality of the product, it should stay fairly consistent. Gameplay shouldn't change drastically, and the game should stay consistent with its goals.

(c) completeness: The product should be complete with most of its main features already implemented and tested. Supplementary features may be implemented, but are not necessary and could be added in the evolution process.

(d) realism: With a stable flow of funds and good budgeting, maintenance of the software is realistic.

(e) verifiability: The requirement is able to be tested. Unity Visual Studios 2015 is great at debugging errors, and with constant feedback from the community, the team should be able to test out future implementations and add them in later.

(f) comprehensibility: The requirement is easily understood as it is a vital part of the entire software process.

(g) traceability: If certain errors should arise, it would be easy to trace back the errors and debug the software. The requirement starts off when the completed software is deployed and evolution occurs.

(h) adaptability: No, the requirement can alter other nonfunctional requirements like overall performance and size.